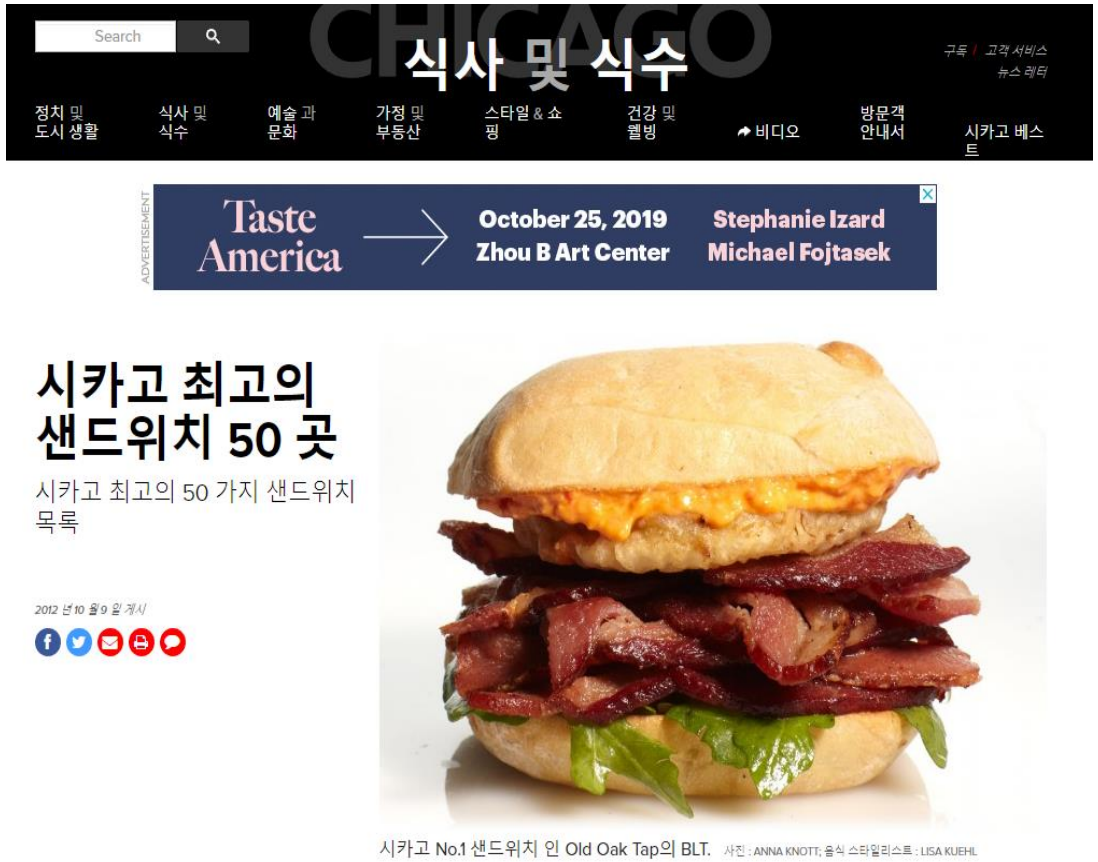
3장 시카고 샌드위치 맛집 분석(125-166쪽)



1. 웹 데이터를 가져오는 Beautiful Soup 익히기
2. 크롬 개발자 도구를 이용해서 원하는 태그 찾기
3. 시카고 샌드위치 맛집 소개 사이트에 접근하기
4. 접근한 웹 페이지에서 원하는 데이터 추출하고 정리하기
5. 다수의 웹 페이지에 자동으로 접근해서 원하는 정보 정리하기
6. Jupyter NoteBook에서 상태 진행바를 쉽게 만들어주는 tqdm모듈
7. 상태 진행바까지 적용하고 다시 샌드위치 페이지 50개에 접근하기
8. 50개 웹 페이지에 대한 정보 가져오기
9. 맛집 위치를 지도에 표기하기
10. 네이버 영화 평점 기준 영화의 평점 변화 확인하기
11. 영화별 날짜 변화에 따른 평점 변화 확인하기
12. 소감

이름 : 구명회
학번 : 20154215
학과 : 컴퓨터 공학과

1. 웹 데이터를 가져오는 Beautiful Soup 익히기
2. 크롬 개발자 도구를 이용해서 원하는 태그 찾기

In [1]:
```
##############################################################################
# 3장 시카고 샌드위치 맛집 분석
# 3-1 Beautiful Soup 라이브러리를 이용해서 인터넷의 정보를 가져오는 작업 수행
# 3-2 메인 페이지와 메인 페이지에 연결된 다른 페이지들의 정보를 읽어옵니다
##############################################################################
# 한 잡지사의 샌드위치 맛집을 소개하는 페이지에서 링크된
# 50개의 페이지에 접속해 원하는 정보를 가져와서 원하는 형태로 정리합니다
# 추가적으로 지도에 각 맛집을 추가합니다
##############################################################################
```

In [2]:
```
# Beautiful Soup bs4를 import
# 웹 데이터 크롤링이나 스크래핑을 할 때 사용하는 Python 라이브러리
# 웹 크롤러 : 인터넷에 있는 웹페이지를 방문해서 자료를 수집하는 일을 하는 프로그램
# 웹 스크래핑 : 컴퓨터 소프트웨어 기술로 웹 사이트들에서 원하는 정보를 추출하는것
# 원래는 Python에서 사용하려면 install 해줘야하는데
# anaconda를 설치할 때 기본적으로 같이 설치됩니다.
from bs4 import BeautifulSoup
```

In [3]:
```
# open()에 읽기 옵션 'r'을 줘서 html 파일을 읽어옵니다
# prettify()를 사용해서 보기좋게 들여쓰기를 해줍니다
page = open("../data/03. test_first.html",'r').read()
soup = BeautifulSoup(page, 'html.parser')
print(soup.prettify())
```

```
<!DOCTYPE html>
<html>
 <head>
  <title>
   Very Simple HTML Code by PinkWink
  </title>
 </head>
 <body>
  <div>
   <p class="inner-text first-item" id="first">
    Happy PinkWink.
    <a href="http://www.pinkwink.kr" id="pw-link">
     PinkWink
    </a>
   </p>
   <p class="inner-text second-item">
    Happy Data Science.
    <a href="https://www.python.org" id="py-link">
     Python
    </a>
   </p>
  </div>
  <p class="outer-text first-item" id="second">
   <b>
    Data Science is funny.
   </b>
  </p>
  <p class="outer-text">
   <b>
    All I need is Love.
   </b>
  </p>
 </body>
</html>
```

In [4]:
```
# children 옵션을 사용해서 전체 html코드를 저장한 변수 soup에서
# 한 단계 아래 포함된 태그들을 확인합니다
list(soup.children)
```

Out[4]:
```
['html', '₩n', <html>
 <head>
 <title>Very Simple HTML Code by PinkWink</title>
 </head>
 <body>
 <div>
 <p class="inner-text first-item" id="first">
                Happy PinkWink.
                <a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>
 </p>
 <p class="inner-text second-item">
                Happy Data Science.
                <a href="https://www.python.org" id="py-link">Python</a>
 </p>
 </div>
 <p class="outer-text first-item" id="second">
 <b>
                Data Science is funny.
            </b>
 </p>
 <p class="outer-text">
 <b>
                All I need is Love.
            </b>
 </p>
 </body>
 </html>]
```

In [5]:
```
# html 태그에 접속합니다
html = list(soup.children)[2]
html
```

Out[5]:
```
<html>
<head>
<title>Very Simple HTML Code by PinkWink</title>
</head>
<body>
<div>
<p class="inner-text first-item" id="first">
                Happy PinkWink.
                <a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>
</p>
<p class="inner-text second-item">
                Happy Data Science.
                <a href="https://www.python.org" id="py-link">Python</a>
</p>
</div>
<p class="outer-text first-item" id="second">
<b>
                Data Science is funny.
            </b>
</p>
<p class="outer-text">
<b>
                All I need is Love.
            </b>
</p>
</body>
</html>
```

In [6]:
```
# html의 children을 조사합니다
list(html.children)
```

Out[6]:
```
['₩n', <head>
 <title>Very Simple HTML Code by PinkWink</title>
 </head>, '₩n', <body>
 <div>
 <p class="inner-text first-item" id="first">
                Happy PinkWink.
                <a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>
 </p>
 <p class="inner-text second-item">
                Happy Data Science.
                <a href="https://www.python.org" id="py-link">Python</a>
 </p>
 </div>
 <p class="outer-text first-item" id="second">
 <b>
                Data Science is funny.
            </b>
 </p>
 <p class="outer-text">
 <b>
                All I need is Love.
            </b>
 </p>
 </body>, '₩n']
```

In [7]:
```
# html의 children중 3번을 조사해서 body의 태그를 확인합니다
body = list(html.children)[3]
body
```

Out[7]:
```
<body>
<div>
<p class="inner-text first-item" id="first">
                Happy PinkWink.
                <a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>
</p>
<p class="inner-text second-item">
                Happy Data Science.
                <a href="https://www.python.org" id="py-link">Python</a>
</p>
</div>
<p class="outer-text first-item" id="second">
<b>
                Data Science is funny.
            </b>
</p>
<p class="outer-text">
<b>
                All I need is Love.
            </b>
</p>
</body>
```

In [8]:
```
# children과 parent를 사용해서 태그를 조사합니다
soup.body
```

Out[8]:
```
<body>
<div>
<p class="inner-text first-item" id="first">
                Happy PinkWink.
                <a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>
</p>
<p class="inner-text second-item">
                Happy Data Science.
                <a href="https://www.python.org" id="py-link">Python</a>
</p>
</div>
<p class="outer-text first-item" id="second">
<b>
                Data Science is funny.
            </b>
</p>
<p class="outer-text">
<b>
                All I need is Love.
            </b>
</p>
</body>
```

In [9]:
```
list(body.children)
```

Out[9]:
```
['₩n', <div>
 <p class="inner-text first-item" id="first">
                Happy PinkWink.
                <a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>
 </p>
 <p class="inner-text second-item">
                Happy Data Science.
                <a href="https://www.python.org" id="py-link">Python</a>
 </p>
 </div>, '₩n', <p class="outer-text first-item" id="second">
 <b>
                Data Science is funny.
            </b>
 </p>, '₩n', <p class="outer-text">
 <b>
                All I need is Love.
            </b>
 </p>, '₩n']
```

In [10]:
```
# body 태그 안의 children 리스트
# 개수를 확인합니다
len(list(body.children))
```

Out[10]:
```
7
```

In [11]:
```
# find_all()를 사용해 접근하려는 태그 p의 모든 태그를 찾습니다
soup.find_all('p')
```

Out[11]:
```
[<p class="inner-text first-item" id="first">
                Happy PinkWink.
                <a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>
 </p>, <p class="inner-text second-item">
                Happy Data Science.
                <a href="https://www.python.org" id="py-link">Python</a>
 </p>, <p class="outer-text first-item" id="second">
 <b>
                Data Science is funny.
            </b>
 </p>, <p class="outer-text">
 <b>
                All I need is Love.
            </b>
 </p>]
```

In [12]:
```
# find()를 사용해 p에 대해 하나의 태그만 찾습니다
soup.find('p')
```

Out[12]:
```
<p class="inner-text first-item" id="first">
                Happy PinkWink.
                <a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>
</p>
```

In [13]:
```
# p 태그의 클래스가 outer-text인 것을 찾습니다
soup.find_all('p', class_='outer-text')
```

Out[13]:
```
[<p class="outer-text first-item" id="second">
 <b>
                Data Science is funny.
            </b>
 </p>, <p class="outer-text">
 <b>
                All I need is Love.
            </b>
 </p>]
```

In [14]:
```
# 클래스 이름으로만 outer-text를 찾습니다
soup.find_all(class_="outer-text")
```

Out[14]:
```
[<p class="outer-text first-item" id="second">
 <b>
                Data Science is funny.
            </b>
 </p>, <p class="outer-text">
 <b>
                All I need is Love.
            </b>
 </p>]
```

In [15]:
```
# id가 first인 태그를 찾습니다
soup.find_all(id="first")
```

Out[15]:
```
[<p class="inner-text first-item" id="first">
                Happy PinkWink.
                <a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>
 </p>]
```

In [16]:
```
# find()는 제일 처음 나타난 태그만 찾습니다
soup.find('p')
```

Out[16]:
```
<p class="inner-text first-item" id="first">
                Happy PinkWink.
                <a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>
</p>
```

In [17]:
```
# head에 있는 내용
soup.head
```

Out[17]:
```
<head>
<title>Very Simple HTML Code by PinkWink</title>
</head>
```

In [18]:
```
# head 다음에 줄바꿈 문자가 있다
soup.head.next_sibling
```

Out[18]:
```
'₩n'
```

In [19]:
```
# head 이전에 줄바꿈 문자가 있다
soup.head.previous_sibling
```

Out[19]:
```
'₩n'
```

In [20]:
```
# head와 같은 위치에 있던 body 태그로 접근합니다
soup.head.next_sibling.next_sibling
```

Out[20]:
```
<body>
<div>
<p class="inner-text first-item" id="first">
                Happy PinkWink.
                <a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>
</p>
<p class="inner-text second-item">
                Happy Data Science.
                <a href="https://www.python.org" id="py-link">Python</a>
</p>
</div>
<p class="outer-text first-item" id="second">
<b>
                Data Science is funny.
            </b>
</p>
<p class="outer-text">
<b>
                All I need is Love.
            </b>
</p>
</body>
```

In [21]:
```
# 제일 처음 나타나는 p태그를 찾습니다
body.p
```

Out[21]:
```
<p class="inner-text first-item" id="first">
                Happy PinkWink.
                <a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>
</p>
```

In [22]:
```
# 제일 처음 나타나는 p태그에 next_sibling을 두 번 걸어
# 그 다음 p태그로 이동했습니다
body.p.next_sibling.next_sibling
```

Out[22]:
```
<p class="inner-text second-item">
                Happy Data Science.
                <a href="https://www.python.org" id="py-link">Python</a>
</p>
```

In [23]:
```
# get_text()를 사용해서 태그 안에 있는
# 텍스트만 출력했습니다
for each_tag in soup.find_all('p'):
    print(each_tag.get_text())
```

```
Happy PinkWink.
PinkWink


Happy Data Science.
Python


Data Science is funny.


All I need is Love.
```

In [24]:
```
# body 전체에서 get_text()를 사용하면 태그가 있던 자리는 줄바꿈(₩n)이 표시되고
# 전체 텍스트를 출력했습니다
body.get_text()
```

Out[24]:
```
'₩n₩n₩n                Happy PinkWink.₩n                PinkWink₩n₩n₩n                Happy Data Science.₩n                Python₩n₩n₩n₩n
₩n                Data Science is funny.₩n                ₩n₩n₩n₩n                All I need is Love.₩n                ₩n₩n'
```

In [25]:
```
# 클릭 가능한 링크를 의미하는 a 태그를 찾습니다
links = soup.find_all('a')
links
```

Out[25]:
```
[<a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>,
 <a href="https://www.python.org" id="py-link">Python</a>]
```

In [26]:
```
# href 속성을 찾아서 링크 주소를 얻었습니다
for each in links:
    href = each['href']
    text = each.string
    print(text + ' -> ' + href)
```

```
PinkWink -> http://www.pinkwink.kr
Python -> https://www.python.org
```

In [27]:
```
# urllib : URL 작업을 위한 모듈을 모아놓은 패키지
# urllib.request : URL을 열고 읽기 위함
# url로 접근하기위해 urlopen함수를 임포트 했습니다
from urllib.request import urlopen
```

In [28]:
```
# 네이버 시장지표 페이지를 읽어와서 출력했습니다
# 현재 페이지에서 접근해야 할 태그를 찾았습니다. >> value
url = "https://finance.naver.com/marketindex/"
page = urlopen(url)

soup = BeautifulSoup(page, "html.parser")

print(soup.prettify())
```

```
<script language="javascript" src="/template/head_js.nhn?referer=info.finance.naver
</script>
<script src="/js/info/jindo.min.ns.1.5.3.euckr.js" type="text/javascript">
</script>
<script src="/js/jindo.1.5.3.element-text-patch.js" type="text/javascript">
</script>
<div id="container" style="padding-bottom:0px;">
 <script language="JavaScript" src="/js/flashObject.js?20190903164201">
 </script>
 <div class="market_include">
  <div class="market_data">
   <div class="market1">
    <div class="title">
     <h2 class="h_market1">
      <span>
       환전 고시 환율
      </span>
```

In [29]:
```
# value 클래스의 0번째 값(태그)을 읽었습니다
soup.find_all('span', 'value')[0].string
```

Out[29]:
```
'1,186.00'
```

1. **BLT**
   Old Oak Tap
   더 읽기

2. **튀김 볼로냐**
   금 슈발
   더 읽기

삼. **우드랜드 버섯**
   Xoco
   더 읽기

4. **로스트 비프**
   알의 델리
   더 읽기

5. **PB & L**
   Publican 고급 육류
   자세히보기

6. **벨기에 치킨 카레 샐러드**
   핸드릭스 (Hendrickx) 벨기에 빵의 장인
   자세히보기

7. **랍스터 롤**
   아카디아
   더 읽기

8. **훈제 연어 샐러드**
   자작 나무 부엌
   더 읽기

9. **Atomica Cemitas**
   Cemitas Puebla
   더 읽기

10. **구이 웃음 조류 새우 튀김 포 '보이**
    나나
    자세히보기

3. 시카고 샌드위치 맛집 소개 사이트에 접근하기

4. 접근한 웹 페이지에서 원하는 데이터 추출하고 정리하기

5. 다수의 웹 페이지에 자동으로 접근해서 원하는 정보 정리하기

6. Jupyter NoteBook에서 상태 진행바를 쉽게 만들어주는 tqdm모듈

7. 상태 진행바까지 적용하고 다시 샌드위치 페이지 50개에 접근하기

8. 50개 웹 페이지에 대한 정보 가져오기

9. 맛집 위치를 지도에 표기하기

<< 시카고 최고의 샌드위치 50곳 중 상위 10개의 맛집 리스트

In [30]:
```
# 전체 html코드를 읽었습니다
from bs4 import BeautifulSoup
from urllib.request import urlopen

# url = 'http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-Chicago/'
url_base = 'http://www.chicagomag.com'
url_sub = '/Chicago-Magazine/November-2012/Best-Sandwiches-Chicago/'
url = url_base + url_sub

html = urlopen(url)
soup = BeautifulSoup(html, "html.parser")

soup
```

Out[30]:

```
<!DOCTYPE doctype html>

<html lang="en">
<head>
<!-- Urbis magnitudo. Fabulas magnitudo. -->
<meta charset="utf-8"/>
<style>a.edit_from_site {display: none !important;}</style>
<title>
  The 50 Best Sandwiches in Chicago |
  Chicago magazine
      |  November 2012
    </title>
<meta content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable = no" name="viewport">
<meta content="Our list of Chicago' s 50 best sandwiches, ranked in order of deliciousness" name="description"/>
<!-- <meta name="description" content="Our list of Chicago' s 50 best sandwiches, ranked in order of deliciousness"> -->
<meta content="sandwiches, dining" name="keywords"/>
<meta content="37873197144" property="fb:pages">
<link href="//www.googletagservices.com" rel="dns-prefetch"/>
```

In [31]:
```
# 확인한 태그를 이용해서 find_all 명령으로 div의 sammy 태그를 찾았습니다
print(soup.find_all('div', 'sammy'))
```

```
[<div class="sammy" style="position: relative;">
<div class="sammyRank">1</div>
<div class="sammyListing"><a href="/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Old-Oak-Tap-BLT/"><b>BLT</b><br>
Old Oak Tap<br>
<em>Read more</em> </br></br></a></div>
</div>, <div class="sammy" style="position: relative;">
<div class="sammyRank">2</div>
<div class="sammyListing"><a href="/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Au-Cheval-Fried-Bologna/"><b>Fried Bologna</
b><br/>
Au Cheval<br/>
<em>Read more</em> </a></div>
</div>, <div class="sammy" style="position: relative;">
<div class="sammyRank">3</div>
<div class="sammyListing"><a href="/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Xoco-Woodland-Mushroom/"><b>Woodland Mushroo
m</b><br/>
Xoco<br/>
<em>Read more</em> </a></div>
</div>, <div class="sammy" style="position: relative;">
<div class="sammyRank">4</div>
<div class="sammyListing"><a href="/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Als-Deli-Roast-Beef/"><b>Roast Beef</b><br/>
Al' s Deli<br/>
```

In [32]:
```
# len 명령어로 길이를 확인했습니다
# 50개의 rank가 있음을 알 수 있습니다
len(soup.find_all('div', 'sammy'))
```

Out[32]:
```
50
```

In [33]:
```
# 그 중에서 첫번째 rank를 확인합니다
print(soup.find_all('div', 'sammy')[0])
```

```
<div class="sammy" style="position: relative;">
<div class="sammyRank">1</div>
<div class="sammyListing"><a href="/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Old-Oak-Tap-BLT/"><b>BLT</b><br>
Old Oak Tap<br>
<em>Read more</em> </br></br></a></div>
</div>
```

In [34]:
```
# sammy 태그에서 원하는 정보를 얻었습니다
# find_all로 찾은 결과는 bs4.element.Tag라는
# 형태라면 변수에 다시 태그로 찾는
# (find, find_all) 명령을 사용할 수 있습니다.
tmp_one = soup.find_all('div', 'sammy')[0]
type(tmp_one)
```

Out [34]:
```
bs4.element.Tag
```

In [35]:
```
# find 명령을 사용해서
# sammyRank를 찾았습니다
tmp_one.find(class_='sammyRank')
```

Out [35]:
```
<div class="sammyRank">1</div>
```

In [36]:
```
# get_text()명령을 사용해서
# sammyRank의 ranking을 구했습니다
tmp_one.find(class_='sammyRank').get_text()
```

Out [36]:
```
'1'
```

In [37]:
```
# sammyListing을 구했습니다
# 메뉴 이름과 가게 이름이 같음을 알 수 있습니다
tmp_one.find(class_='sammyListing').get_text()
```

Out [37]:
```
'BLT₩r₩nOld Oak Tap₩nRead more '
```

In [38]:
```
# a태그에서 href 정보를 가지고 클릭했을 때 연결될 주소를 저장합니다
tmp_one.find('a')['href']
```

Out [38]:
```
'/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Old-Oak-Tap-BLT/'
```

In [39]:
```
# sammyListing에서 메뉴 이름과 가게 이름이 같기 때문에
# 이것을 따로 분리할 필요가 있습니다
# 분리하기 위해 정규식을 사용하려고 합니다
# 정규식을 쓰기 위해 re를 임포트했습니다
# split는 지정한 특정 패턴이 일치하면 분리시킵니다
# 첫번째는 메뉴이름으로, 두번째는 가게이름으로 분리시켰습니다
import re

tmp_string = tmp_one.find(class_='sammyListing').get_text()

re.split(('₩n|₩r₩n'), tmp_string)

print(re.split(('₩n|₩r₩n'), tmp_string)[0])
print(re.split(('₩n|₩r₩n'), tmp_string)[1])
```

```
BLT
Old Oak Tap
```

In [40]:
```
# 절대경로로 잡힌 url은 그대로 두고
# 상대경로로 잡힌 url은 절대 경로로 변경합니다.
from urllib.parse import urljoin
```

In [41]:
```
# rank :랭크 순위
# main_menu : 메인 메뉴 이름
# cafe_name : 카페 이름
# url_add : 각각의 접근 주소
# for문을 돌려 append 명령어를 사용해
# find_all('div', 'sammy')로 찾은 50개의 정보를
# 비어있는 리스트에 추가했습니다
rank = []
main_menu = []
cafe_name = []
url_add = []

list_soup = soup.find_all('div', 'sammy')

for item in list_soup:
    rank.append(item.find(class_='sammyRank').get_text())

    tmp_string = item.find(class_='sammyListing').get_text()

    main_menu.append(re.split(('₩n|₩r₩n'), tmp_string)[0])
    cafe_name.append(re.split(('₩n|₩r₩n'), tmp_string)[1])

    url_add.append(urljoin(url_base, item.find('a')['href']))
```

In [42]:
```
# 상태바를 사용해서 현재 진행상태를 출력했습니다
# 나머지 코드는 위의 코드와 똑같습니다.
from tqdm import tqdm_notebook
import time

rank = []
main_menu = []
cafe_name = []
url_add = []

list_soup = soup.find_all('div', 'sammy')
bar_total = tqdm_notebook(list_soup)

for item in bar_total:
    rank.append(item.find(class_='sammyRank').get_text())

    tmp_string = item.find(class_='sammyListing').get_text()

    main_menu.append(re.split(('₩n|₩r₩n'), tmp_string)[0])
    cafe_name.append(re.split(('₩n|₩r₩n'), tmp_string)[1])

    url_add.append(urljoin(url_base, item.find('a')['href']))

    time.sleep(0.05)
```

In [43]:
```
# 순위
rank[:5]
```

Out[43]:
```
['1', '2', '3', '4', '5']
```

In [44]:
```
# 메뉴 이름
main_menu[:5]
```

Out[44]:
```
['BLT', 'Fried Bologna', 'Woodland Mushroom', 'Roast Beef', 'PB&L']
```

In [45]:
```
# 카페 이름
cafe_name[:5]
```

Out[45]:
```
['Old Oak Tap', 'Au Cheval', 'Xoco', 'Al' s Deli', 'Publican Quality Meats']
```

In [46]:
```
# 접근 경로(url)
url_add[:5]
```

Out[46]:
```
['http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Old-Oak-Tap-BLT/',
 'http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Au-Cheval-Fried-Bologna/',
 'http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Xoco-Woodland-Mushroom/',
 'http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Als-Deli-Roast-Beef/',
 'http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Publican-Quality-Meats-PB-L/']
```

In [47]:
```
# 각각의 길이
# 50개의 순위, 메뉴 이름, 카페이름, url
len(rank), len(main_menu), len(cafe_name), len(url_add)
```

Out[47]:
```
(50, 50, 50, 50)
```

In [48]:
```
# pandas를 사용해서 자료를 데이터 프레임으로 정리했습니다
import pandas as pd

data = {'Rank':rank, 'Menu':main_menu, 'Cafe':cafe_name, 'URL':url_add}
df = pd.DataFrame(data)
df.head()
```

Out [48]:

	Rank	Menu	Cafe	URL
0	1	BLT	Old Oak Tap	http://www.chicagomag.com/Chicago-Magazine/Nov...
1	2	Fried Bologna	Au Cheval	http://www.chicagomag.com/Chicago-Magazine/Nov...
2	3	Woodland Mushroom	Xoco	http://www.chicagomag.com/Chicago-Magazine/Nov...
3	4	Roast Beef	Al's Deli	http://www.chicagomag.com/Chicago-Magazine/Nov...
4	5	PB&L	Publican Quality Meats	http://www.chicagomag.com/Chicago-Magazine/Nov...

In [49]:
```
# 컬럼의 순서를 재배치 했습니다
df = pd.DataFrame(data, columns=['Rank','Cafe','Menu','URL'])
df.head(5)
```

Out [49]:

	Rank	Cafe	Menu	URL
0	1	Old Oak Tap	BLT	http://www.chicagomag.com/Chicago-Magazine/Nov...
1	2	Au Cheval	Fried Bologna	http://www.chicagomag.com/Chicago-Magazine/Nov...
2	3	Xoco	Woodland Mushroom	http://www.chicagomag.com/Chicago-Magazine/Nov...
3	4	Al's Deli	Roast Beef	http://www.chicagomag.com/Chicago-Magazine/Nov...
4	5	Publican Quality Meats	PB&L	http://www.chicagomag.com/Chicago-Magazine/Nov...

In [50]:
```
# 데이터를 저장했습니다
df.to_csv('../data/03. best_sandwiches_list_chicago.csv', sep=',',
          encoding='UTF-8')
```

In [51]:
```
# Beautiful Soup bs4를 import
# 웹 데이터 크롤링이나 스크래핑에 사용하는 Python 라이브러리
# 원래는 Python에서 사용하려면 install 해줘야하는데
# anaconda를 설치할 때 기본적으로 같이 설치됩니다.
# urllib : URL 작업을 위한 모듈을 모아놓은 패키지
# urllib.request : URL을 열고 읽기 위함
# url로 접근하기위해 urlopen함수를 임포트
# 데이터 프레임으로 정리하기 위해 pandas를 임포트
from bs4 import BeautifulSoup
from urllib.request import urlopen

import pandas as pd
```

In [52]:
```
# 이전에 저장했던 데이터 프레임을 읽어왔습니다
df = pd.read_csv('../data/03. best_sandwiches_list_chicago.csv', index_col=0)
df.head()
```

Out [52]:

	Rank	Cafe	Menu	URL
0	1	Old Oak Tap	BLT	http://www.chicagomag.com/Chicago-Magazine/Nov...
1	2	Au Cheval	Fried Bologna	http://www.chicagomag.com/Chicago-Magazine/Nov...
2	3	Xoco	Woodland Mushroom	http://www.chicagomag.com/Chicago-Magazine/Nov...
3	4	Al's Deli	Roast Beef	http://www.chicagomag.com/Chicago-Magazine/Nov...
4	5	Publican Quality Meats	PB&L	http://www.chicagomag.com/Chicago-Magazine/Nov...

In [53]:
```
# rank의 첫번째 url을 읽어옵니다.
df['URL'][0]
```

Out [53]:
```
'http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Old-Oak-Tap-BLT/'
```

In [54]:
```
# rank의 첫번째 주소를 Beautiful Soup로 읽었습니다
html = urlopen(df['URL'][0])
soup_tmp = BeautifulSoup(html, "html.parser")
soup_tmp
```

Out[54]:

```
<!DOCTYPE doctype html>

<html lang="en">
<head>
<!-- Urbis magnitudo. Fabulas magnitudo. -->
<meta charset="utf-8"/>
<style>a.edit_from_site {display: none !important;}</style>
<title>
  1. Old Oak Tap BLT |
  Chicago magazine
      |  November 2012
    </title>
<meta content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable = no" name="viewport">
<meta content=" “Truly inspired.” " name="description"/>
<!-- <meta name="description" content="&#8220;Truly inspired.&#8221;"> -->
<meta content="" name="keywords"/>
<meta content="37873197144" property="fb:pages">
<link href="//www.googletagservices.com" rel="dns-prefetch"/>
<link href="//ajax.googleapis.com" rel="dns-prefetch"/>
<link href="//securepubads.g.doubleclick.net" rel="dns-prefetch"/>
<link href="//media.chicagomag.com" rel="dns-prefetch"/>
<link href="//ox-d.godengo.com/" rel="dns-prefetch"/>
<link href="//www.google-analytics.com" rel="dns-prefetch"/>
<link href="//ping.chartbeat.net" rel="dns-prefetch"/>
<link href="//static.chartbeat.com" rel="dns-prefetch"/>
<link href="//www.google.com" rel="dns-prefetch"/>
<link href="//cse.google.com" rel="dns-prefetch"/>
<link href="//www.googleapis.com" rel="dns-prefetch"/>
<link href="//maps.googleapis.com" rel="dns-prefetch"/>
<link href="//csi.gstatic.com" rel="dns-prefetch"/>
<link href="//www.facebook.com" rel="dns-prefetch"/>
<link href="//staticxx.facebook.com" rel="dns-prefetch"/>
```

In [55]:
```
# p클래스의 addy태그를 읽었습니다
print(soup_tmp.find('p', 'addy'))
```

```
<p class="addy">
<em>$10. 2109 W. Chicago Ave., 773-772-0406, <a href="http://www.theoldoaktap.com/">theoldoaktap.com</a></em></p>
```

In [56]:
```
# get_text()로 태그의 내용을 읽어왔습니다
price_tmp = soup_tmp.find('p', 'addy').get_text()
price_tmp
```

Out[56]:
```
'₩n$10. 2109 W. Chicago Ave., 773-772-0406, theoldoaktap.com'
```

In [57]:
```
# split는 지정한 특정 패턴이 일치하면 분리시킵니다
# 가격,
price_tmp.split()
```

Out[57]:
```
['$10.', '2109', 'W.', 'Chicago', 'Ave.,', '773-772-0406,', 'theoldoaktap.com']
```

In [58]:
```
# 가격
price_tmp.split()[0]
```

Out[58]:
```
'$10.'
```

In [59]:
```
# '.'을 제거하고 출력합니다
# price_tmp.split()[0][:3]
# '', '$', '$1', '$10', '$10.'
price_tmp.split()[0][:-1]
```

Out[59]:
```
'$10'
```

In [60]:
```
# price_tmp.split() 리스트에서
# price_tmp[1] ~ price_tmp[6-2] 까지
# 출력합니다
' '.join(price_tmp.split()[1:-2])
```

Out[60]:
```
'2109 W. Chicago Ave.,'
```

In [61]:
```
# price : 가격 순위
# address : 각각의 접근 주소
# for문을 돌려 append 명령어를 사용해
# find_all('p', 'addy')로 찾은 3개의 정보를
# 비어있는 리스트에 추가했습니다
price = []
address = []

for n in df.index[:3]:
    html = urlopen(df['URL'][n])
    soup_tmp = BeautifulSoup(html, 'lxml')

    gettings = soup_tmp.find('p', 'addy').get_text()

    price.append(gettings.split()[0][:-1])
    address.append(' '.join(gettings.split()[1:-2]))
```

In [62]:
```
# 가격
price
```

Out[62]:
```
['$10', '$9', '$9.50']
```

In [63]:
```
# 주소
address
```

Out[63]:
```
['2109 W. Chicago Ave.,', '800 W. Randolph St.,', '445 N. Clark St.,']
```

In [64]:
```
# 상태바를 사용해서 현재 진행상태를 출력했습니다
# 50개 하위 페이지에 대한 가격과 주소를 얻었습니다
from tqdm import tqdm_notebook

price = []
address = []

for n in tqdm_notebook(df.index):
    html = urlopen(df['URL'][n])
    soup_tmp = BeautifulSoup(html, 'lxml')

    gettings = soup_tmp.find('p', 'addy').get_text()

    price.append(gettings.split()[0][:-1])
    address.append(' '.join(gettings.split()[1:-2]))
```


100% 50/50 [05:06<00:00, 5.95s/it]

In [65]:
```
# 가격을 출력합니다
# price,, 50개 리스트
# 그중에 3개만 출력합니다
price[:3]
```

Out [65]:
```
['$10', '$9', '$9.50']
```

In [66]:
```
# 주소를 출력합니다
# address,, 50개 리스트
# 그중에 6개만 출력합니다
address[:6]
```

Out [66]:
```
['2109 W. Chicago Ave.,',
 '800 W. Randolph St.,',
 '445 N. Clark St.,',
 '914 Noyes St., Evanston,',
 '825 W. Fulton Mkt.,',
 '100 E. Walton']
```

In [67]:
```
# 가격, 주소, 웹 페이지 정보의 길이
len(price), len(address), len(df)
```

Out [67]:
```
(50, 50, 50)
```

In [68]:
```
# 웹 페이지 정보에 가격과 주소를 추가했습니다
# 새로 추가한 가격과 주소의 순서를 재배치하고
# rank를 인덱스로 설정해서 출력했습니다
df['Price'] = price
df['Address'] = address

df = df.loc[:, ['Rank', 'Cafe', 'Menu', 'Price', 'Address']]
df.set_index('Rank', inplace=True)
df.head()
```

Out [68]:

Rank	Cafe	Menu	Price	Address
1	Old Oak Tap	BLT	$10	2109 W. Chicago Ave.,
2	Au Cheval	Fried Bologna	$9	800 W. Randolph St.,
3	Xoco	Woodland Mushroom	$9.50	445 N. Clark St.,
4	Al's Deli	Roast Beef	$9.40	914 Noyes St., Evanston,
5	Publican Quality Meats	PB&L	$10	825 W. Fulton Mkt.,

In [69]:
```
# 지금까지 결과 저장
df.to_csv('../data/03. best_sandwiches_list_chicago2.csv', sep=',',
          encoding='UTF-8')
```

In [70]:
```
# pip install folium >>> folium 라이브러리 설치
# 지도에 맛집 표현하기 위해 시각화 도구 folium 임포트
# google 지도를 사용하기위해 googlemaps 임포트
# 자료를 데이터 프레임으로 정리하기 위해 pandas 임포트
# 행렬 개념으로 데이터를 표현하기 위해 numpy 임포트
import folium
import googlemaps
import pandas as pd
import numpy as np
```

In [71]:
```
# 저장했던 데이터를 불러옵니다
df = pd.read_csv('../data/03. best_sandwiches_list_chicago2.csv', index_col=0)
df.head(5)
```

Out[71]:

Rank	Cafe	Menu	Price	Address
1	Old Oak Tap	BLT	$10	2109 W. Chicago Ave.,
2	Au Cheval	Fried Bologna	$9	800 W. Randolph St.,
3	Xoco	Woodland Mushroom	$9.50	445 N. Clark St.,
4	Al's Deli	Roast Beef	$9.40	914 Noyes St., Evanston,
5	Publican Quality Meats	PB&L	$10	825 W. Fulton Mkt.,

In [72]:
```
# "googlemaps에서 획득한 고유 라이센스 키"
gmaps_key = "AIzaSyDjqdtQBjvQVHYI3bWENsC7qU7v77nLI44"
gmaps = googlemaps.Client(key=gmaps_key)
```

In [73]:
```
# 50개 맛집의 위도와 경도 정보를 받아옵니다
# Multiple이 나타나지 않는 경우만 주소를 검색합니다
# 상태바를 사용해서 진행상태를 출력합니다
lat = []
lng = []

for n in tqdm_notebook(df.index):
    if df['Address'][n] != 'Multiple':
        target_name = df['Address'][n]+', '+'Chicago'
        gmaps_output = gmaps.geocode(target_name)
        location_output = gmaps_output[0].get('geometry')
        lat.append(location_output['location']['lat'])
        lng.append(location_output['location']['lng'])

    else:
        lat.append(np.nan)
        lng.append(np.nan)
```

100% 50/50 [00:19<00:00, 2.44it/s]

In [74]:
```
# 50개 맛집의 위도와
# 경도를 저장했으므로
# 각각 50개씩 출력됩니다
len(lat), len(lng)
```

Out [74]:
```
(50, 50)
```

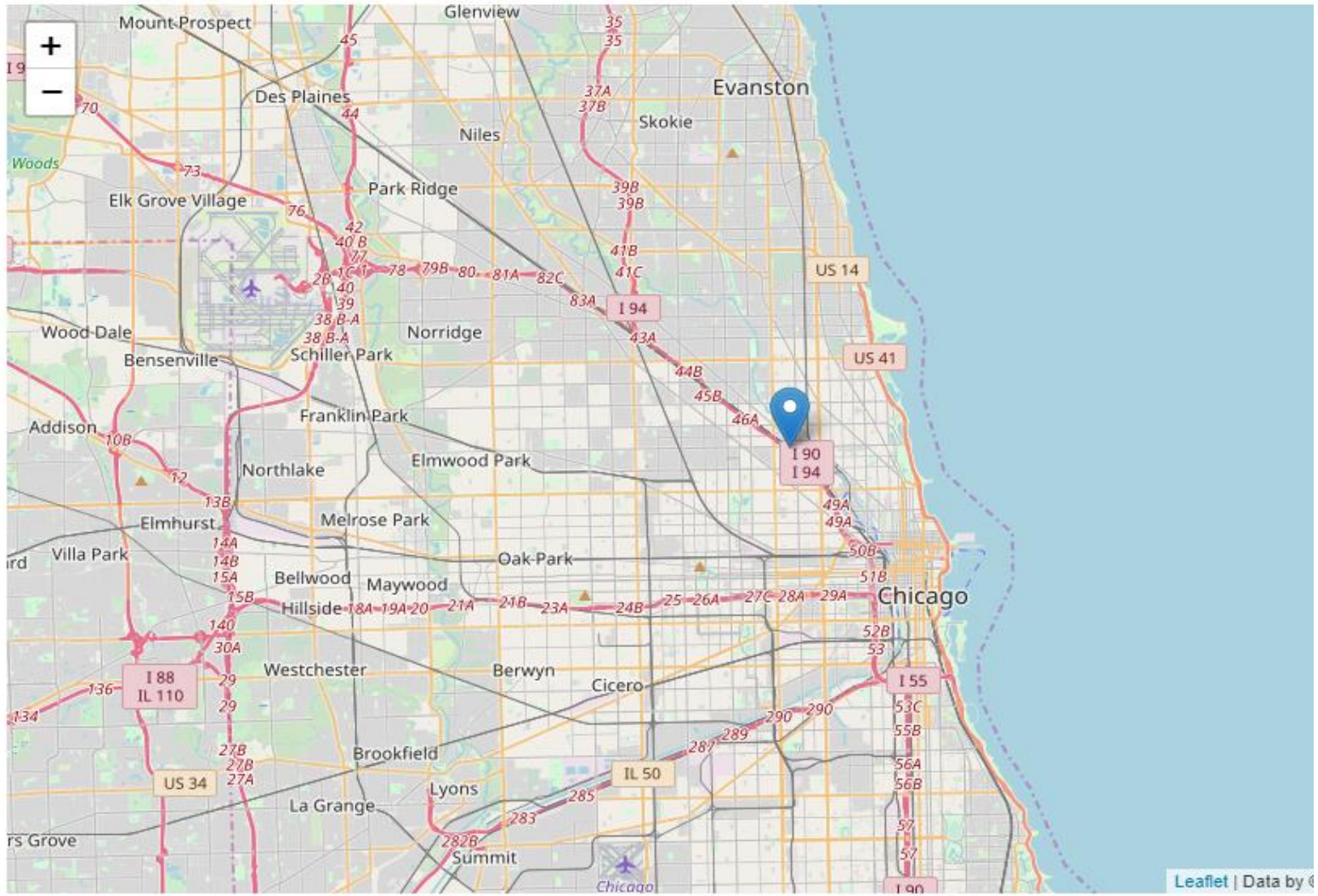
In [75]:
```
# 웹 페이지 정보에 위도와 경도를 추가하고
# 웹 페이지 정보를 출력했습니다
df['lat'] = lat
df['lng'] = lng
df.head()
```

Out [75]:

Rank	Cafe	Menu	Price	Address	lat	lng
1	Old Oak Tap	BLT	$10	2109 W. Chicago Ave.,	41.895605	-87.679961
2	Au Cheval	Fried Bologna	$9	800 W. Randolph St.,	41.884658	-87.647667
3	Xoco	Woodland Mushroom	$9.50	445 N. Clark St.,	41.890618	-87.630933
4	Al's Deli	Roast Beef	$9.40	914 Noyes St., Evanston,	42.058322	-87.683748
5	Publican Quality Meats	PB&L	$10	825 W. Fulton Mkt.,	41.886600	-87.648451

In [76]:
```
# 50개 맛집의 위도와 경도의 평균값을 중앙으로 둡니다
mapping = folium.Map(location=[df['lat'].mean(), df['lng'].mean()],
                                    zoom_start=11)
folium.Marker([df['lat'].mean(), df['lng'].mean()],
                                    popup='center').add_to(mapping)
mapping
```

Out [76]:

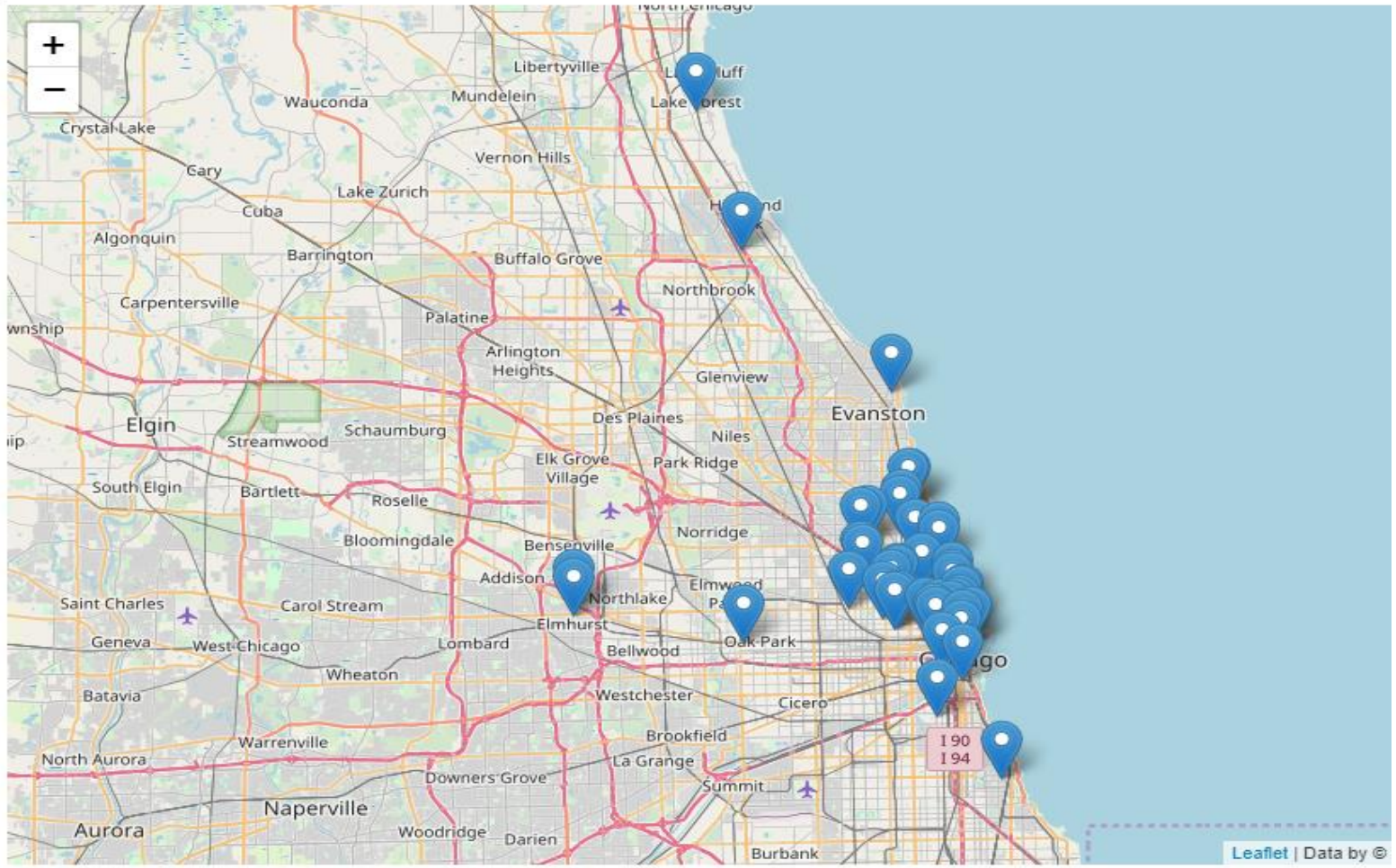


In [77]:
```
# 50개 맛집의 위도와 경도를 지도에 표시했습니다
mapping = folium.Map(location=[df['lat'].mean(), df['lng'].mean()],
                     zoom_start=11)

for n in df.index:
    if df['Address'][n] != 'Multiple':
        folium.Marker([df['lat'][n], df['lng'][n]],
                                    popup=df['Cafe'][n]).add_to(mapping)

mapping
```

Out [77]:



영화 랭킹

조회순 | 평점순 (현재상영영화) | 평점순 (모든영화) | 2019.10.12

순위	영화명	평점		변동폭
1	주전장	9.51	평점주기	– 0
2	사랑의 선물	9.44	평점주기	– 0
3	안녕 베일리	9.43	평점주기	– 0
4	아이언 자이언트	9.33	평점주기	– 0
5	폴란드로 간 아이들	9.32	평점주기	– 0
6	언더독	9.30	평점주기	– 0
7	교회오빠	9.28	평점주기	– 0
8	우리들	9.25	평점주기	– 0
9	레드슈즈	9.24	평점주기	– 0
10	벌새	8.94	평점주기	– 0
11	우리집	8.78	평점주기	↑ 1
12	소공녀	8.78	평점주기	↑ 1
13	조커	8.78	평점주기	↓ 2
14	기생충	8.48	평점주기	– 0
15	양자물리학	8.39	평점주기	– 0
16	장사리 : 잊혀진 영웅들	8.39	평점주기	– 0
17	메기	8.31	평점주기	NEW!
18	엑시트	8.30	평점주기	↓ 1
19	죽거나 혹은 나쁘거나	8.07	평점주기	↓ 1
20	퍼펙트맨	8.03	평점주기	– 0
21	가장 보통의 연애	8.02	평점주기	↓ 2
22	시인의 사랑	7.98	평점주기	↓ 1
23	더 룸	7.98	평점주기	↓ 1
24	에곤 쉴레: 욕망이 그린 그림	7.92	평점주기	↓ 1
25	장기왕: 가락시장 레볼루션	7.82	평점주기	↓ 1
26	콜 미 바이 유어 네임	7.69	평점주기	↓ 1
27	원스 어폰 어 타임... 인 할리우드	7.64	평점주기	↓ 1
28	힘을 내요, 미스터 리	7.57	평점주기	↓ 1
29	극장판 헬로카봇 : 달나라를 구해줘!	7.49	평점주기	↓ 1
30	판소리 복서	7.36	평점주기	NEW!

영화 인기검색어 ▸더보기

1	조커	– 0
2	가장 보통의 연애	– 0
3	퍼펙트맨	– 0
4	제미니 맨	– 0
5	양자물리학	↑ 1

2019.10.12

영화인 인기검색어 ▸더보기

1	호아킨 피닉스	– 0
2	최유화	↑ 1
3	토드 필립스	↓ 1
4	임지연	↑ 2
5	재지 비츠	– 0

2019.10.12

티켓예매순 ▸더보기

1	조커	49.84%
2	가장 보통의 연애	28.12%
3	퍼펙트맨	7.65%
4	제미니 맨	7.16%
5	소피와 드래곤: 마..	1.27%

출처:YES24 2019.10.13

박스오피스 ▸더보기

1	조커	1,285,880 명
2	가장 보통의 연애	624,854 명
3	퍼펙트맨	325,462 명
4	장사리 : 잊혀진 …	102,350 명
5	소피와 드래곤: …	35,800 명

주말관객 기준 20191004-20191006

10. 네이버 영화 평점 기준 영화의 평점 변화 확인하기

11. 영화별 날짜 변화에 따른 평점 변화 확인하기

<< 2019.10.12 네이버 영화 랭킹 중 상위 30위의 영화 리스트

In [1]:
```
# Beautiful Soup bs4를 import
# 웹 데이터 크롤링이나 스크래핑을 할 때 사용하는 Python 라이브러리
# 웹 크롤러 : 인터넷에 있는 웹페이지를 방문해서 자료를 수집하는 일을 하는 프로그램
# 웹 스크래핑 : 컴퓨터 소프트웨어 기술로 웹 사이트들에서 원하는 정보를 추출하는것
# 원래는 Python에서 사용하려면 install 해줘야하는데
# anaconda를 설치할 때 기본적으로 같이 설치됩니다.
# 자료를 데이터 프레임으로 정리하기 위해 pandas 임포트
from bs4 import BeautifulSoup
import pandas as pd
```

In [2]:
```
# urllib : URL 작업을 위한 모듈을 모아놓은 패키지
# urllib.request : URL을 열고 읽기 위함
# url로 접근하기위해 urlopen함수를 임포트
# url_syb = "movie/sdb/rank/rmovie.nhn?sel=cur&date=20170804"
# 에서 현재 날짜 20191012로 수정했습니다.
from urllib.request import urlopen

url_base = "http://movie.naver.com/"
url_syb = "movie/sdb/rank/rmovie.nhn?sel=cur&date=20191012"

page = urlopen(url_base+url_syb)

soup = BeautifulSoup(page, "html.parser")
soup
```

Out[2]:

```
<!DOCTYPE html>

<html lang="ko">
<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type"/>
<meta content="IE=edge" http-equiv="X-UA-Compatible"/>
<meta content="http://imgmovie.naver.com/today/naverme/naverme_profile.jpg" property="me2:image">
<meta content="네이버영화 " property="me2:post_tag">
<meta content="네이버영화" property="me2:category1"/>
<meta content="" property="me2:category2"/>
<meta content="랭킹 : 네이버 영화" property="og:title"/>
<meta content="영화, 영화인, 예매, 박스오피스 랭킹 정보 제공" property="og:description"/>
<meta content="article" property="og:type"/>
<meta content="https://movie.naver.com/movie/sdb/rank/rmovie.nhn?sel=cur&amp;date=20191012" proper
<meta content="http://static.naver.net/m/movie/icons/OG_270_270.png" property="og:image"/><!-- htt
ie.jpg -->
<meta content="http://imgmovie.naver.com/today/naverme/naverme_profile.jpg" property="og:article:t
<meta content="네이버 영화" property="og:article:author"/>
<meta content="https://movie.naver.com/" property="og:article:author:url"/>
```

In [3]:
```
# div클래스의 tit5 태그를 읽었습니다
soup.find_all('div', 'tit5')
```

Out[3]:
```
[<div class="tit5">
 <a href="/movie/bi/mi/basic.nhn?code=179518" title="주전장">주전장</a>
 </div>, <div class="tit5">
 <a href="/movie/bi/mi/basic.nhn?code=182699" title="사랑의 선물">사랑의 선물</a>
 </div>, <div class="tit5">
 <a href="/movie/bi/mi/basic.nhn?code=181700" title="안녕 베일리">안녕 베일리</a>
 </div>, <div class="tit5">
 <a href="/movie/bi/mi/basic.nhn?code=25915" title="아이언 자이언트">아이언 자이언트</a>
 </div>, <div class="tit5">
 <a href="/movie/bi/mi/basic.nhn?code=178434" title="폴란드로 간 아이들">폴란드로 간 아이들</a>
 </div>, <div class="tit5">
 <a href="/movie/bi/mi/basic.nhn?code=144318" title="언더독">언더독</a>
 </div>, <div class="tit5">
 <a href="/movie/bi/mi/basic.nhn?code=183132" title="교회오빠">교회오빠</a>
 </div>, <div class="tit5">
 <a href="/movie/bi/mi/basic.nhn?code=146504" title="우리들">우리들</a>
 </div>, <div class="tit5">
 <a href="/movie/bi/mi/basic.nhn?code=164907" title="레드슈즈">레드슈즈</a>
 </div>, <div class="tit5">
 <a href="/movie/bi/mi/basic.nhn?code=179307" title="벌새">벌새</a>
 </div>, <div class="tit5">
 <a href="/movie/bi/mi/basic.nhn?code=185450" title="우리집">우리집</a>
 </div>, <div class="tit5">
 <a href="/movie/bi/mi/basic.nhn?code=159311" title="소공녀">소공녀</a>
 </div>, <div class="tit5">
 <a href="/movie/bi/mi/basic.nhn?code=167613" title="조커">조커</a>
 </div>, <div class="tit5">
 <a href="/movie/bi/mi/basic.nhn?code=161967" title="기생충">기생충</a>
 </div>, <div class="tit5">
 <a href="/movie/bi/mi/basic.nhn?code=187629" title="양자물리학">양자물리학</a>
```

In [4]:
```
# 태그의 첫번째를 출력합니다
soup.find_all('div', 'tit5')[0]
```

Out[4]:
```
<div class="tit5">
<a href="/movie/bi/mi/basic.nhn?code=179518" title="주전장">주전장</a>
</div>
```

In [5]:
```
# 첫번째 태그에서 제목이 포함된 라인만 출력합니다
soup.find_all('div', 'tit5')[0].a
```

Out[5]:
```
<a href="/movie/bi/mi/basic.nhn?code=179518" title="주전장">주전장</a>
```

In [6]:
```
# 첫번째 태그에서 제목만 출력합니다
soup.find_all('div', 'tit5')[0].a.string
```

Out[6]:
```
'주전장'
```

In [7]:
```
# 첫번째 태그의 평점들을 출력합니다
soup.find_all('td', 'point')
```

Out[7]:
```
[<td class="point">9.51</td>,
 <td class="point">9.44</td>,
 <td class="point">9.43</td>,
 <td class="point">9.33</td>,
 <td class="point">9.32</td>,
 <td class="point">9.30</td>,
 <td class="point">9.28</td>,
 <td class="point">9.25</td>,
 <td class="point">9.24</td>,
 <td class="point">8.94</td>,
 <td class="point">8.78</td>,
 <td class="point">8.78</td>,
 <td class="point">8.78</td>,
 <td class="point">8.48</td>,
```

In [8]:
```
# 첫번째 태그의 평점 갯수를 출력합니다
len(soup.find_all('td', 'point'))
```

Out[8]:
```
40
```

In [9]:
```
# 첫번째 태그의 첫번째 평점을 출력합니다
soup.find_all('td', 'point')[0].string
```

Out[9]:
```
'9.51'
```

In [10]:
```
# 40개의 영화 이름을 출력합니다
movie_name = [soup.find_all('div', 'tit5')[n].a.string for n in range(0, 40)]
movie_name
```

Out[10]:
```
['주전장',
 '사랑의 선물',
 '안녕 베일리',
 '아이언 자이언트',
 '폴란드로 간 아이들',
 '언더독',
 '교회오빠',
 '우리들',
 '레드슈즈',
 '벌새',
 '우리집',
 '소공녀',
 '조커',
 '기생충',
 '양자물리학',
```

In [11]:
```
# 40개의 영화 평점을 출력합니다
movie_point = [soup.find_all('td', 'point')[n].string for n in range(0, 40)]
movie_point
```

Out[11]:
```
['9.51',
 '9.44',
 '9.43',
 '9.33',
 '9.32',
 '9.30',
 '9.28',
 '9.25',
 '9.24',
 '8.94',
 '8.78',
 '8.78',
```

In [12]:
```
# 날짜를 20190912부터 한달(30일)간으로 정의했습니다
date = pd.date_range('2019-9-12', periods=30, freq='D')
date
```

Out[12]:
```
DatetimeIndex(['2019-09-12', '2019-09-13', '2019-09-14', '2019-09-15',
               '2019-09-16', '2019-09-17', '2019-09-18', '2019-09-19',
               '2019-09-20', '2019-09-21', '2019-09-22', '2019-09-23',
               '2019-09-24', '2019-09-25', '2019-09-26', '2019-09-27',
               '2019-09-28', '2019-09-29', '2019-09-30', '2019-10-01',
               '2019-10-02', '2019-10-03', '2019-10-04', '2019-10-05',
               '2019-10-06', '2019-10-07', '2019-10-08', '2019-10-09',
               '2019-10-10', '2019-10-11'],
              dtype='datetime64[ns]', freq='D')
```

In [13]:
```
# urllib : URL 작업을 위한 모듈을 모아놓은 패키지
# 상태바를 사용해서 현재 진행상태를 출력합니다
# movie_date : 영화 날짜
# movie_name : 영화 이름
# movie_point : 영화 평점
import urllib
from tqdm import tqdm_notebook

movie_date = []
movie_name = []
movie_point = []

for today in tqdm_notebook(date):
    html = "http://movie.naver.com/" + ₩
                                    "movie/sdb/rank/rmovie.nhn?sel=cur&date={date}"
    response = urlopen(html.format(date=
                                   urllib.parse.quote(today.strftime('%Y%m%d'))))
    soup = BeautifulSoup(response, "html.parser")

    end = len(soup.find_all('td', 'point'))

    movie_date.extend([today for n in range(0, end)])
    movie_name.extend([soup.find_all('div', 'tit5')[n].a.string for n in range(0, end)])
    movie_point.extend([soup.find_all('td', 'point')[n].string for n in range(0, end)])
```

In [14]:
```
# 영화의 날짜, 이름, 평점의 길이를 출력합니다
len(movie_date), len(movie_name), len(movie_point)
```

Out[14]:
```
(1126, 1126, 1126)
```

In [15]:
```
# 영화의 날짜, 이름, 평점을 pandas를 사용해
# 데이터 프레임으로 저장한 뒤 출력했습니다
movie = pd.DataFrame({'date':movie_date, 'name':movie_name,
                      'point':movie_point})
movie.head()
```

Out[15]:

	date	name	point
0	2019-09-12	안녕 베일리	9.55
1	2019-09-12	주전장	9.54
2	2019-09-12	알라딘	9.39
3	2019-09-12	집으로...	9.34
4	2019-09-12	메리 포핀스	9.27

In [16]:
```
# ..
movie.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1126 entries, 0 to 1125
Data columns (total 3 columns):
date     1126 non-null datetime64[ns]
name     1126 non-null object
point    1126 non-null object
dtypes: datetime64[ns](1), object(2)
memory usage: 26.5+ KB
```

In [17]:
```
# ..
movie['point'] = movie['point'].astype(float)
movie.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1126 entries, 0 to 1125
Data columns (total 3 columns):
date     1126 non-null datetime64[ns]
name     1126 non-null object
point    1126 non-null float64
dtypes: datetime64[ns](1), float64(1), object(1)
memory usage: 26.5+ KB
```

In [18]:
```
# 피벗 테이블을 사용해서 영화별 점수를 합산한 데이터를 출력합니다
# 영화별 점수의 합계로 정렬하기 위해 aggfunc의 np.sum옵션을 이용합니다
# 9월 12일부터 한달(30일)간 점수를 합산했을때 고득점을 받은 영화를
# 1위부터 5위까지 출력합니다
import numpy as np

movie_unique = pd.pivot_table(movie, index=['name'], aggfunc=np.sum)
movie_best = movie_unique.sort_values(by='point', ascending=False)
movie_best.head()
```

Out[18]:

name	point
주전장	286.18
안녕 베일리	283.65
교회오빠	278.19
벌새	268.73
우리집	264.06

In [19]:
```
# 조커의 날짜별 평점 변화를 확인했습니다
tmp = movie.query('name == ["조커"]')
tmp
```

Out[19]:

	date	name	point
784	2019-10-02	조커	9.27
820	2019-10-03	조커	9.12
856	2019-10-04	조커	9.05
893	2019-10-05	조커	8.98
933	2019-10-06	조커	8.91
967	2019-10-07	조커	8.87
1000	2019-10-08	조커	8.86
1033	2019-10-09	조커	8.83
1064	2019-10-10	조커	8.82
1098	2019-10-11	조커	8.80

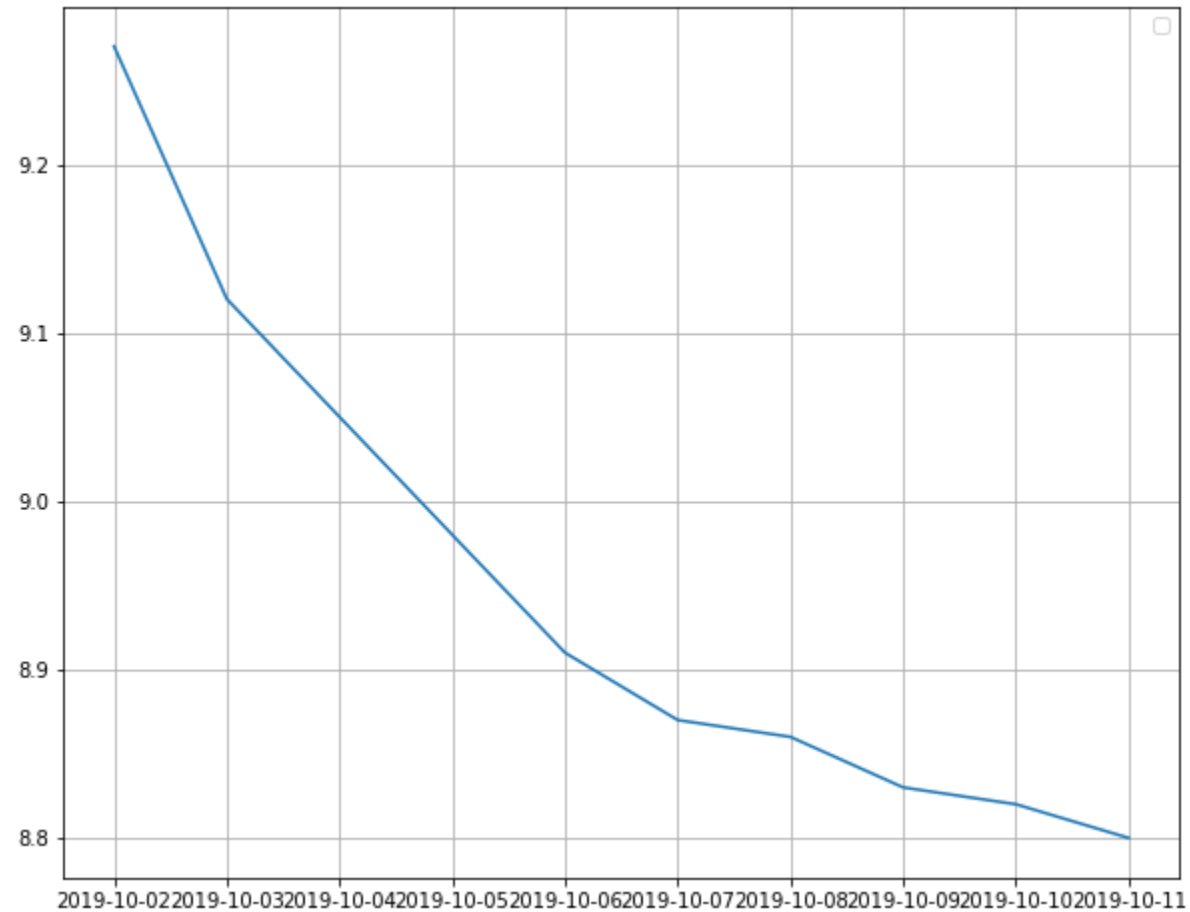
In [20]:

```
# %matplotlib inline : notebook을 실행한 브라우저에서
# 바로 그림을 볼 수 있게 해줍니다.
# 날짜별로 그래프를 그렸습니다
# 상위 버전에서는 legend의 옵션으로 handle을 구현해줘야 합니다
import matplotlib.pyplot as plt
%matplotlib inline

plt.figure(figsize=(10,8))
plt.plot(tmp['date'], tmp['point'])
plt.legend(loc='best')
plt.grid()
plt.show()
```

```
C:₩Users₩nerin₩Anaconda3₩lib₩site-packages₩pandas₩plotting₩_converter.p
rter for a matplotlib plotting method. The converter was registered by
licitly register matplotlib converters.

To register the converters:
        >>> from pandas.plotting import register_matplotlib_converters
        >>> register_matplotlib_converters()
  warnings.warn(msg, FutureWarning)
No handles with labels found to put in legend.
```



In [21]:
```
# 날짜별로 정리된 데이터를 피벗 테이블을 사용해
# 가로에는 영화제목을, 세로에는 날짜를 넣어 정리해 출력했습니다
movie_pivot = pd.pivot_table(movie, index=["date"], columns=['name'], values=['point'])
movie_pivot.head()
```

Out[21]:

	point																			
name	47 미터 2	가장 보통의 연애	건축학개론	광대들: 풍문조작단	교회 오빠	그것: 두 번째 이야기	그녀	그린 북	극장판 헬로카봇 : 달나라를 구해줘!	기생충	...	커런트 워	콜 미 바이 유어 네임	킹 오브 프리즘 -샤이니 세븐 스타즈-	타샤 튜더	타짜: 원 아이드 잭	토이 스토리 4	퍼스트맨	퍼펙트맨	폴란드로 간 아이들	힘을 내요, 미스터리
date																					
2019-09-12	7.26	NaN	8.65	6.19	9.26	6.22	NaN	NaN	7.50	8.48	...	7.81	NaN	NaN	9.03	5.73	NaN	NaN	NaN	NaN	7.92
2019-09-13	7.27	NaN	8.65	6.20	9.26	6.21	NaN	NaN	7.58	8.48	...	7.80	NaN	NaN	9.03	5.50	NaN	NaN	NaN	NaN	7.83
2019-09-14	7.28	NaN	8.65	6.21	9.26	6.22	NaN	NaN	7.56	8.48	...	7.82	NaN	NaN	9.03	5.42	NaN	NaN	NaN	NaN	7.76
2019-09-15	7.28	NaN	8.65	6.21	9.26	6.22	NaN	NaN	7.55	8.48	...	7.82	NaN	NaN	9.03	5.43	NaN	NaN	NaN	NaN	7.76
2019-09-16	7.28	NaN	NaN	6.22	9.27	6.22	NaN	NaN	7.57	8.48	...	7.82	NaN	NaN	9.03	5.40	9.09	NaN	NaN	NaN	7.70

5 rows × 85 columns

In [22]:
```
# error!
# 인터넷에 검색해봐도 무슨 문장인지 잘 모르겠습니다..
movie_pivot.columns = movie_pivot.columns.droplevel()
```

In [23]:
```
# 하지만 column.droplevel을 보면
# 첫번째 col(point)을 삭제하는것..이라고 유추했습니다
movie_pivot.head()
```

Out [23]:

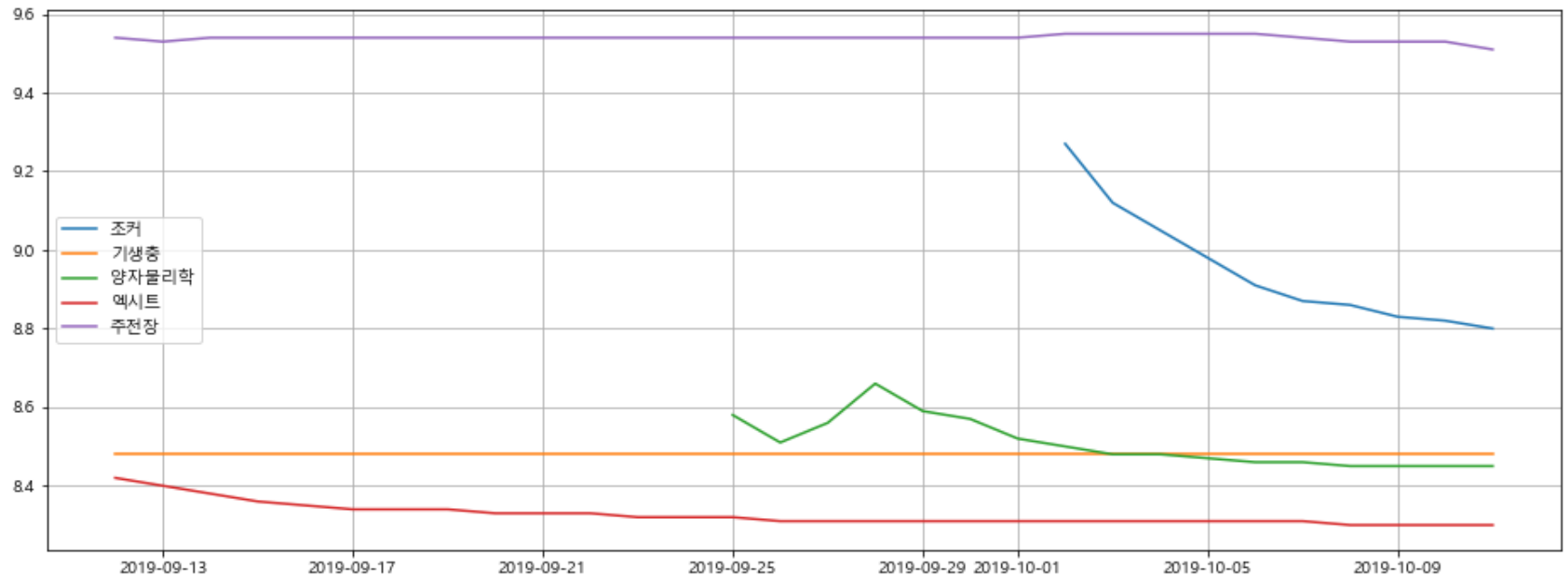
name	47 미터 2	가장 보통의 연애	건축 학개론	광대들: 풍문조작단	교회 오빠	그것: 두 번째 이야기	그녀	그린 북	극장판 헬로카봇 : 달나라를 구해줘!	기생충	...	커런트 워	콜 미 바이 유어 네임	킹 오브 프리즘 -샤이니 세븐 스타즈-	타샤 튜더	타짜: 원 아이드 잭	토이 스토리 4	퍼스트맨	퍼펙트맨	폴란드로 간 아이들	힘을 내요, 미스터리
date																					
2019-09-12	7.26	NaN	8.65	6.19	9.26	6.22	NaN	NaN	7.50	8.48	...	7.81	NaN	NaN	9.03	5.73	NaN	NaN	NaN	NaN	7.92
2019-09-13	7.27	NaN	8.65	6.20	9.26	6.21	NaN	NaN	7.58	8.48	...	7.80	NaN	NaN	9.03	5.50	NaN	NaN	NaN	NaN	7.83
2019-09-14	7.28	NaN	8.65	6.21	9.26	6.22	NaN	NaN	7.56	8.48	...	7.82	NaN	NaN	9.03	5.42	NaN	NaN	NaN	NaN	7.76
2019-09-15	7.28	NaN	8.65	6.21	9.26	6.22	NaN	NaN	7.55	8.48	...	7.82	NaN	NaN	9.03	5.43	NaN	NaN	NaN	NaN	7.76
2019-09-16	7.28	NaN	NaN	6.22	9.27	6.22	NaN	NaN	7.57	8.48	...	7.82	NaN	NaN	9.03	5.40	9.09	NaN	NaN	NaN	7.70

5 rows × 85 columns

In [24]:
```
# 피벗 테이블의 결과에 붙은 컬럼 제목을 정리합니다
# matplotlib에서 한글 문제를 설정했습니다
import platform
from matplotlib import font_manager, rc

path = "c:/Windows/Fonts/malgun.ttf"
if platform.system() == 'Darwin':
    rc('font', family='AppleGothic')
elif platform.system() == 'Windows':
    font_name = font_manager.FontProperties(fname=path).get_name()
    rc('font', family=font_name)
else:
    print('Unknown system... sorry~~~~')
```

In [25]:
```
# 관심있는 영화 5개를 선택해서 날짜별 변화를 확인했습니다
target_col = [ '조커', '기생충', '양자물리학', '엑시트', '주전장']
plt.figure(figsize=(16,6))
plt.plot(movie_pivot[target_col])
plt.legend(target_col, loc='best')
plt.grid()
plt.show()
```



12. 소감

이번 과제에서는 html 언어의 해석과 웹 크롤링/스크래핑에 대해서 조금 알게 되었습니다.
지금까지는 인터넷 검색을 하다가 가끔 F12키를 잘못 눌렀을 때 html언어가 나오면
잘못 눌렀다고 짜증내면서 끄기 바빴는데 이번엔 컴퓨터를 만지게 된 이래 처음으로
과제 덕분에.. HTML언어에 관심을 가지고 구조를 살펴보게 되었습니다.
특히 이번 과제 3-10, 11에서 올해 영화 리스트에 대해 정보를 가지고 있는 링크에서
웹 크롤링/스크래핑 해온 데이터를 분석할 수 있어서 재미있었습니다.

아쉬웠던 점은 책에 명령어를 왜 사용하는지 설명이 잘 되어있지 않았습니다.
인터넷에 검색해보면 대부분 나오지만 책처럼 두리뭉실하게 설명된 부분이 많아서
변수나 상수를 바꿔가며 이해하는데 시간이 많이 걸렸습니다.